JUNIT API

Important API's of JUnit

The most important package in JUnit is **junit.framework** which contain all the core classes. Some of the important class are

Serial No	Class Name	Functionality
1	Assert	A set of assert methods.
2	TestCase	A test case defines the fixture to run multiple tests.
3	TestResult	A TestResult collects the results of executing a test case.
4	TestSuite	A TestSuite is a Composite of Tests.

Assert Class

Following is the declaration for **org.junit.Assert** class:

```
public class Assert extends java.lang.Object
```

This class provides a set of assertion methods useful for writing tests. Only failed assertions are recorded. Some of the important methods of **Assert** class are:

S.N.	Methods & Description
1	**void assertEquals(boolean expected, boolean actual)** Check that two primitives/Objects are equal
2	**void assertFalse(boolean condition)** Check that a condition is false
3	**void assertNotNull(Object object)** Check that an object isn't null.
4	**void assertNull(Object object)** Check that an object is null
5	**void assertTrue(boolean condition)** Check that a condition is true.
6	**void fail()** Fails a test with no message.

Let's try to cover few of the above mentioned methods in an example. Create a java class file name TestJunit1.java in **C:\ > JUNIT_WORKSPACE**

```
import org.junit.Test;
import static org.junit.Assert.*;
public class TestJunit1 {
   @Test
   public void testAdd() {
      //test data
      int num= 5;
      String temp= null;
      String str= "Junit is working fine";

      //check for equality
      assertEquals("Junit is working fine", str);

      //check for false condition
      assertFalse(num > 6);

      //check for not null value
      assertNotNull(str);
   }
}
```

Next, let's create a java class file name TestRunner1.java in **C:\ > JUNIT_WORKSPACE** to execute Test case(s)

```
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;

public class TestRunner1 {
   public static void main(String[] args) {
      Result result = JUnitCore.runClasses(TestJunit1.class);
      for (Failure failure : result.getFailures()) {
         System.out.println(failure.toString());
      }
      System.out.println(result.wasSuccessful());
   }
}
```

Compile the Test case and Test Runner classes using javac

```
C:\JUNIT_WORKSPACE>javac TestJunit1.java TestRunner1.java
```

Now run the Test Runner which will run test case defined in provided Test Case class.

```
C:\JUNIT_WORKSPACE>java TestRunner1
```

Verify the output.

```
true
```

TestCase Class

Following is the declaration for **org.junit.TestCaset** class:

```
public abstract class TestCase extends Assert implements Test
```

A test case defines the fixture to run multiple tests. Some of the important methods of **TestCase** class are

S.N.	Methods & Description
1	**int countTestCases()**

	Counts the number of test cases executed by run(TestResult result).
2	**TestResult createResult()** Creates a default TestResult object.
3	**String getName()** Gets the name of a TestCase.
4	**TestResult run()** A convenience method to run this test, collecting the results with a default TestResult object.
5	**void run(TestResult result)** Runs the test case and collects the results in TestResult.
6	**void setName(String name)** Sets the name of a TestCase.
7	**void setUp()** Sets up the fixture, for example, open a network connection.
8	**void tearDown()** Tears down the fixture, for example, close a network connection.
9	**String toString()** Returns a string representation of the test case.

Let's try to cover few of the above mentioned methods in an example. Create a java class file name TestJunit2.java in **C:\ > JUNIT_WORKSPACE**

```
import junit.framework.TestCase;
import org.junit.Before;
import org.junit.Test;
public class TestJunit2 extends TestCase  {
   protected double fValue1;
   protected double fValue2;

   @Before
   public void setUp() {
      fValue1= 2.0;
      fValue2= 3.0;
   }

   @Test
   public void testAdd() {
      //count the number of test cases
      System.out.println("No of Test Case = "+ this.countTestCases());

      //test getName
      String name= this.getName();
      System.out.println("Test Case Name = "+ name);

      //test setName
      this.setName("testNewAdd");
      String newName= this.getName();
      System.out.println("Updated Test Case Name = "+ newName);
   }
   //tearDown used to close the connection or clean up activities
   public void tearDown(  ) {
   }
}
```

Next, let's create a java class file name TestRunner2.java in **C:\ > JUNIT_WORKSPACE** to execute Test case(s)

```
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;

public class TestRunner2 {
   public static void main(String[] args) {
      Result result = JUnitCore.runClasses(TestJunit2.class);
      for (Failure failure : result.getFailures()) {
         System.out.println(failure.toString());
      }
      System.out.println(result.wasSuccessful());
   }
}
```

Compile the Test case and Test Runner classes using javac

```
C:\JUNIT_WORKSPACE>javac TestJunit2.java TestRunner2.java
```

Now run the Test Runner which will run test case defined in provided Test Case class.

```
C:\JUNIT_WORKSPACE>java TestRunner2
```

Verify the output.

```
No of Test Case = 1
Test Case Name = testAdd
Updated Test Case Name = testNewAdd
true
```

TestResult Class

Following is the declaration for **org.junit.TestResult** class:

```
public class TestResult extends Object
```

A TestResult collects the results of executing a test case. It is an instance of the Collecting Parameter pattern. The test framework distinguishes between failures and errors. A failure is anticipated and checked for with assertions. Errors are unanticipated problems like an ArrayIndexOutOfBoundsException. Some of the important methods of **TestResult** class are

S.N.	Methods & Description
1	**void addError(Test test, Throwable t)** Adds an error to the list of errors.
2	**void addFailure(Test test, AssertionFailedError t)** Adds a failure to the list of failures.
3	**void endTest(Test test)** Informs the result that a test was completed.
4	**int errorCount()** Gets the number of detected errors.
5	**Enumeration<TestFailure> errors()**

	Returns an Enumeration for the errors.
6	**int failureCount()** Gets the number of detected failures.
7	**void run(TestCase test)** Runs a TestCase.
8	**int int runCount()** Gets the number of run tests.
9	**void startTest(Test test)** Informs the result that a test will be started.
10	**void stop()** Marks that the test run should stop.

Create a java class file name TestJunit3.java in **C:\ > JUNIT_WORKSPACE**

```
import org.junit.Test;
import junit.framework.AssertionFailedError;
import junit.framework.TestResult;

public class TestJunit3 extends TestResult {
   // add the error
   public synchronized void addError(Test test, Throwable t) {
      super.addError((junit.framework.Test) test, t);
   }

   // add the failure
   public synchronized void addFailure(Test test, AssertionFailedError t) {
      super.addFailure((junit.framework.Test) test, t);
   }
   @Test
   public void testAdd() {
   // add any test
   }

   // Marks that the test run should stop.
   public synchronized void stop() {
   //stop the test here
   }
}
```

Next, let's create a java class file name TestRunner3.java in **C:\ > JUNIT_WORKSPACE** to execute Test case(s)

```
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;

public class TestRunner3 {
   public static void main(String[] args) {
      Result result = JUnitCore.runClasses(TestJunit3.class);
      for (Failure failure : result.getFailures()) {
         System.out.println(failure.toString());
      }
      System.out.println(result.wasSuccessful());
   }
}
```

Compile the Test case and Test Runner classes using javac

```
C:\JUNIT_WORKSPACE>javac TestJunit3.java TestRunner3.java
```

Now run the Test Runner which will run test case defined in provided Test Case class.

```
C:\JUNIT_WORKSPACE>java TestRunner3
```

Verify the output.

```
true
```

TestSuite Class

Following is the declaration for **org.junit.TestSuite** class:

```
public class TestSuite extends Object implements Test
```

A TestSuite is a Composite of Tests. It runs a collection of test cases. Some of the important methods of **TestSuite** class are

S.N.	Methods & Description
1	**void addTest(Test test)** Adds a test to the suite.
2	**void addTestSuite(Class<? extends TestCase> testClass)** Adds the tests from the given class to the suite.
3	**int countTestCases()** Counts the number of test cases that will be run by this test.
4	**String getName()** Returns the name of the suite.
5	**void run(TestResult result)** Runs the tests and collects their result in a TestResult.
6	**void setName(String name)** Sets the name of the suite.
7	**Test testAt(int index)** Returns the test at the given index.
8	**int testCount()** Returns the number of tests in this suite.
9	**static Test warning(String message)** Returns a test which will fail and log a warning message.

Create a java class file name JunitTestSuite.java in **C:\ > JUNIT_WORKSPACE** to create Test suite

```
import junit.framework.*;
public class JunitTestSuite {
   public static void main(String[] a) {
      // add the test's in the suite
      TestSuite suite = new TestSuite(TestJunit1.class,
      TestJunit2.class, TestJunit3.class );
      TestResult result = new TestResult();
      suite.run(result);
      System.out.println("Number of test cases = " + result.runCount());
    }
}
```

Compile the Test suite classes using javac

```
C:\JUNIT_WORKSPACE>javac JunitTestSuite.java
```

Now run the Test Suite.

```
C:\JUNIT_WORKSPACE>java JunitTestSuite
```

Verify the output.

```
No of Test Case = 1
Test Case Name = testAdd
Updated Test Case Name = testNewAdd
Number of test cases = 3
```